Evaluating and Controlling Transformation Variants

Ian Karlin (U of Colorado; summer student at Argonne)
Jean Utke(Argonne, MCS)

- preaccumulation, fixed graphs
- taping, checkpointing
- preaccumulation variable graphs

why do we worry about this?

- one goal of OpenAD is easy implementation of transformations
- now we have a number of variations to the some themes
- don’t know the effect of selecting a particular set until we run it
- need to evaluate and (semi)-automatically pick the “best”

preaccumulation

have (fixed) DAG, want Jacobian entries ... in OpenAD:

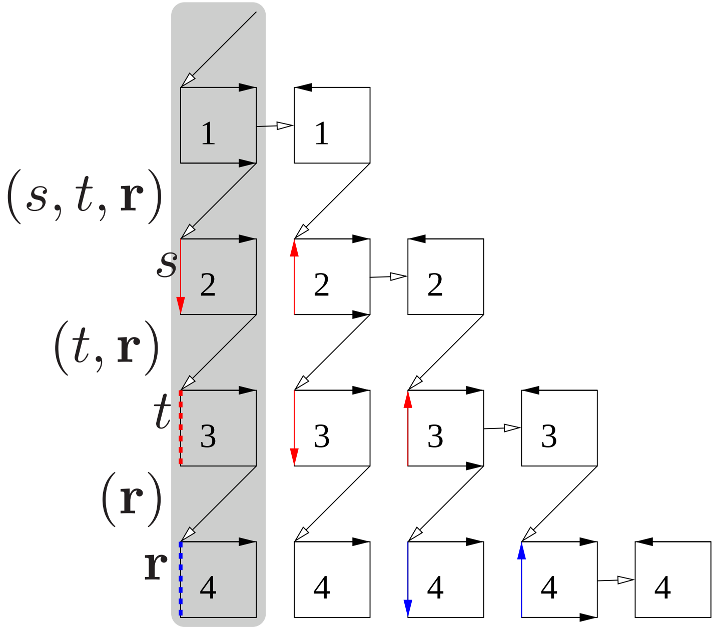
- angel
  - vertex, edge, face elimination
  - heuristics (lowest (relat.) Mark., lowest fill, lowest Mark. minimal damage,..)
  - heuristic sequence, compares varying sequences based on number of fma
  - comparison to LSA (vertex, face)
- mem-ops tradeoff (data locality vs operations count)
  - vertex, edge, face elimination
  - Markowitz, Absorption, forw., rev.
  - sibling, parent-child , etc. specific for data locality, didn't work well ☹
- fma, better mult. and add. separate, (simple measure for locality)
- collected, compared → automatic selection Simple ☺

checkpointing 1

- jump to a “high-level” consideration - placement of checkpoints
- in OpenAD: pick a subroutine, sideffect analysis determines the checkpoint
- this is for the entire call subtree (if non-recursive)
- how big is the checkpoint?
- static estimates: number of scalars, vectors, matrices etc. w/o dimension
- primary goal is to aid the user with a model for checkpoints
- assumes some insight (e.g. the size of that 5-tensor is X)
- static information available for all subroutines in the call graph (Ian Karlin)

checkpointing 2

- dynamic estimates: count them for a scaled down example
- a bit more difficult
- start with building the dynamic call tree (vertex for each actual call)
- experiment yields detailed but unwieldy presentation
- could also look at xaif in browser
- →collapse to call graph
- next practical problem was data visibility

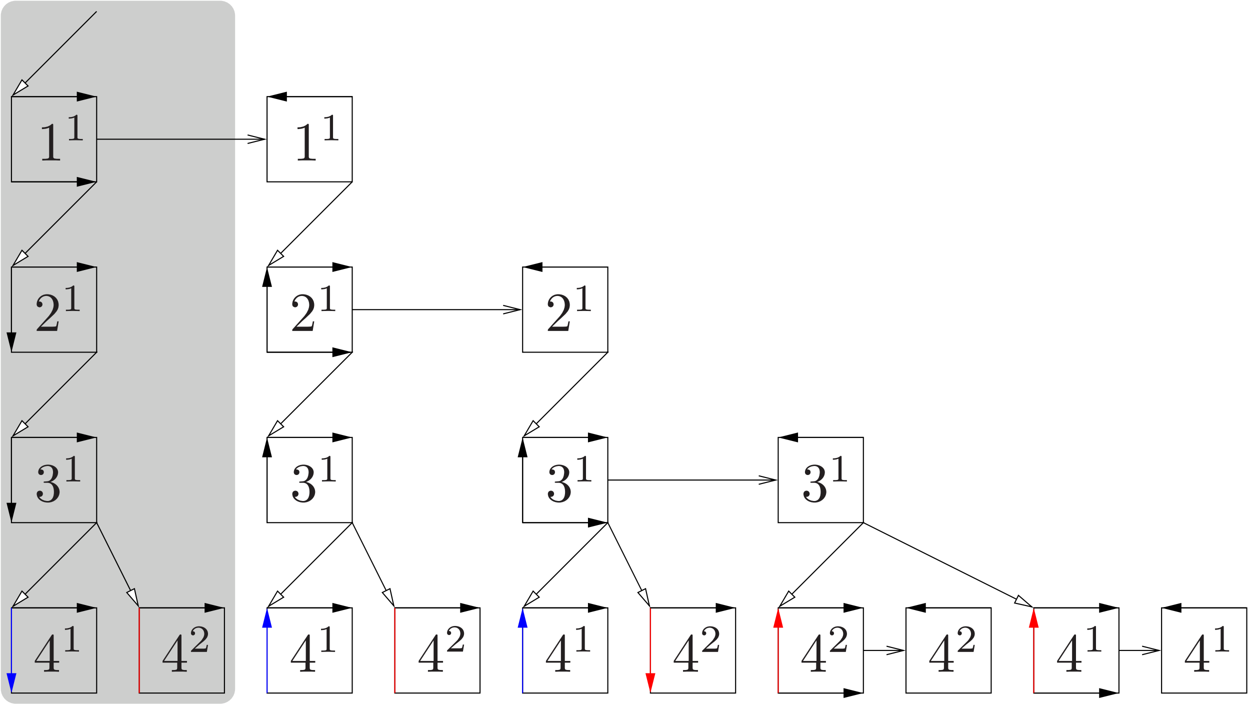


- can't see **r** in 2 or 3
- make it visible → have naming conflicts

rename or insert a cp subroutine or use dynamic call tree?

checkpointing 3

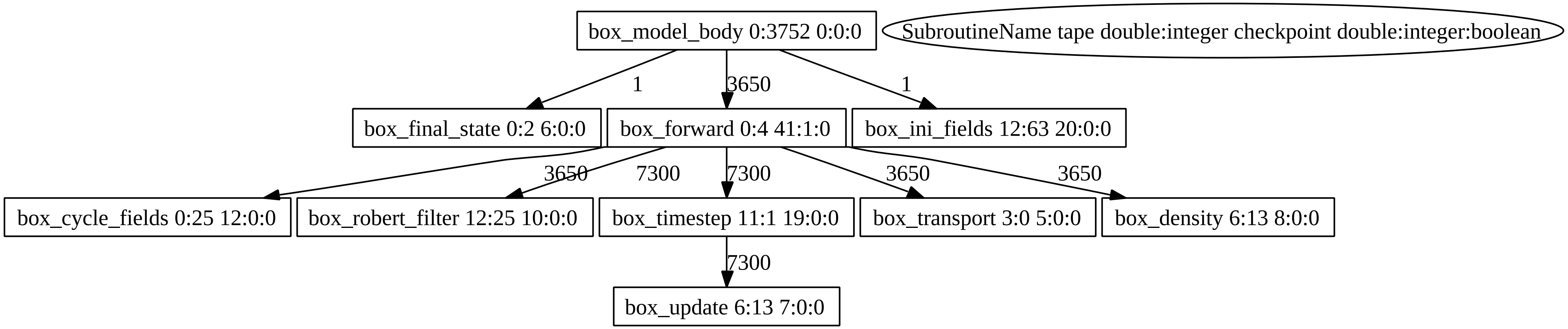
with the dynamic call tree



- dynamic call tree for reducing cp write counts (Nice 2005)
- less rigid scheme (unify tape&checkpoints Laurent H. in Reading)
- need some more practical tests

taping

- does the tape fit?
- in OpenAD: preaccumulation
  → tape Jacobian entries
- dynamic data size, loop bounds
  → profile using counters per subroutine
- given as node annotations in call graph



box_model_body 0:3752 0:0:0

SubroutineName tape double:integer checkpoint double:integer:boolean

1

3650

1

box_final_state 0:2 6:0:0

box_forward 0:4 41:1:0

box_ini_fields 12:63 20:0:0

3650

7300

7300

3650

3650

box_cycle_fields 0:25 12:0:0

box_robert_filter 12:25 10:0:0

box_timestep 11:1 19:0:0

box_transport 3:0 5:0:0

box_density 6:13 8:0:0

7300

box_update 6:13 7:0:0

... obviously this still needs some work to be useful

preaccumulation 2

what happens if we pick “smaller” DAGs?

- so far in OpenAD: maximize DAG within aliasing and control flow limits
- total effort:
  - varying cost for local preaccumulation (elimination order)
  - fixed cost for global propagation,
    i.e. taping space and/or sparse matrix-vector products
- plausible expectation: choose smaller (more) DAGs
  → effort shifts from preaccumulation to propagation
- but in some cases: lower number of preaccumulation ops *and* fewer Jacobian entries (totaled per basic block)
  - in maximized DAGs the last elimination are expensive
  - using smaller DAGs means allowing “incomplete” preaccumulations

preaccumulation 3

IOW we suspect scarsity.

- do not consider a nested problem (NP-hard optimal elimination within varying DAG splits)
- use the maximal DAG and apply a scarsity preserving heuristic
- more complicated code for matrix-vector products
- does not consider rerouting

summary

- more arguments for the DCT approach to CP placement
- generic use of scarsity concept
- useful transformation variations require local comparisons and decisions
- preaccumulation heuristics consistent with runtime results
- does not address the bigger question:
  "to preaccumulate or not to preaccumulate..."
  - is a local question
  - see also: *"to store or recompute"*
  - basic block is the natural scope

AD and Source Transformation Systems

STS 2006 Portland

- target various applications, also complex transformations
- even the holy grail, aka C++ refactoring
- many systems under active development
  - stratego, tom, elan etc...
  - Holland, France
- problems with "acceptance"
  - lack of *good* front-ends & analysis tools
  - interest in pooling resources
- hope to get an STS overview presentation for the next Euro AD workshop